International Journal of Computer Science Engineering
and Information Technology Research(IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 5, Issue 2, April 2015, 59-70
@TJPRC Pvt. Ltd.

ENHANCEMENT OF SEMANTIC MATCHMAKING IN WEB SERVICES

SAVAN M. PATEL[1] & GAURAV K. PATEL[2]

[1]Department of CE, U. V. Patel Engineering College, Kherva, Gujarat, India

[2]Department of CSE, LCIT College, Bhandu, Gujarat, India

ABSTRACT

The ability to dynamically discover and invoke a Web Service is a critical aspect of Service Oriented Architectures. An important component of the discovery process is the matchmaking algorithm itself. In order to overcome the limitations of a syntax-based search, matchmaking algorithms based on semantic techniques have been proposed. As a greater number of Web Services are made available, support for service discovery mechanisms become essential. So we have use the semantic matchmaking using the greedy approach, that work on the greedy concept. The reault shows that the matchmaking with greedy concept gets the fast requested service as compare to the other algorithms. Matchmaking plays a vital role in this discovery process.

KEYWORDS: Web Service Composition, Ontology, Web Services, Semantic Web Services. Semantic Similarity

INTRODUCTION

With contributions from Semantic Web technology, Web services are enhanced using rich description languages such as Web Ontology Language [1]. Semantic Web Services [2] are Web services that have been enhanced with formal semantic descriptions in Web Ontology Language of Web services [3]. OWL-S is an OWL-based Web services ontology that provides Web service suppliers with a core set of markup languages that is constructed for describing properties and functions, and the grounding information for Web services in a computer-interpretable form.

As each service is added to constructed composition workflow, a parameter matching mechanism is necessary to make sure that the service supports the **Inputs**, **Outputs**, **Preconditions**, and **Effects (I/O/P/E)** constraints of the workflow. In addition, I/O/P/E parameters of the selected services affect the matching step for the next node of the constituted workflow. Therefore, in order to obtain proper compositions of Web services, a powerful matching process is required. Including potential services in a composition task is critical for the assessment of the composition.

The Semantic Web approaches to web services give us the ability to describe the semantics of web services and their capabilities in a formal and machine-process manner. The Semantic Web should enable greater access not only to content but also to services on the web. Users and software agents should be able to discover, invoke, compose, and monitor web resources offering particular services and having particular properties, and should be able to do so with a high degree of automation if desired. The focus of this paper is on using the semantic meta data which is published for web services for measuring the similarity of web services. The ability of measuring the similarity of web services has several use cases. The following is a brief description of some of these use cases:

- **Service Discovery:** Semantic web service discovery or matchmaking is an emerging research area that exploits

the service semantic metadata to locate services that can perform tasks of given description with the best over all degree of satisfaction. Most of the proposed service discovery algorithms are based on measuring the similarity of the requested service with each of the advertised services. Therefore having a similarity measure for web services can be quite helpful in service discovery.

- **Service Recommendation:** It is often happens that a client spent some time for manually finding a service that can fulfil his requirements by browsing categories in a registry and then try to invoke the service but the service is unavailable. In this situation, similar services can be suggested to the client.
- **Service Composition:** Currently, many services are offered on the web and their number would increase exponentially in the future. But these atomic services may not be able to satisfy specific requests individually. So there must be brokers between service providers and service requesters which integrate the atomic services of service providers and create new value added services. The act of taking several services and bundling them together to meet the needs of a given customer is called service composition.

BACKGROUND AND RELATED WORK

Ontologies are used in order to incorporate semantics in web service descriptions. An Ontology models domain knowledge in terms of Concepts and Relationships between them. OWL[12] [11] has evolved as a standard for representation of ontologies on the Web. OWL-S[18] [11], formerly called DAML-S[10], defines an ontology for semantic web services. OWL-S describes a service in terms of its Service Profile, Process Model and Grounding. The Service Profile models the Inputs, Outputs,Pre-conditions and Effects (IOPE) of the service. The Inputs and Outputs in the Service Profile refer to concepts in ontologies published on theWeb. Service Advertisements and search Queries are both expressed in terms of OWL-S descriptions.

An ontology reasoner is an important component in the process of semantic matchmaking. A reasoner can infer additional information which has not been explicitly stated in an ontology. Subsumption, concept satisfiability, equivalence and disjointness are some examples of reasoning operations. Many of these operations are used in the semantic matchmaking process. DAML-S is based on a logic formalism called Description Logics (DL). Description Logics and its reasoning are explored in detail by[15] and[19]. Racer[7] and Pellet[5] [23] are some implementations of DL-Reasoners. The Service Profile contains enough information for a matchmaker to determine if a service satisfies the requirements of a client. In fact, several matchmaking algorithms rely only on the matching of Inputs and Outputs of the Service Profiles.

SEMANTIC MATCHMAKING ALGORITHM

The algorithm takes a OWLS *Query* from the client as input and iterates over every OWL-S *Advertisement* in its repository in order to determine a match. An *Advertisement* and a *Query* match if their Outputs and Inputs, both, match. The algorithm returns a set of matching advertisements sorted according to the degree of match.

Let Queryout and Advtout represent the *list of output concepts* of the *Query* and *Advertisement* respectively. Matching the outputs requires the matching between two *concept-lists*, Queryout and Advtout, as follows:

$\forall c \in$ Queryout, $\exists d \in$ Advtout,

s.t. match(c, d) != Fail

Let Queryin and Advtin represent the list of input Concepts of the Query and Advertisement respectively. Matching the inputs requires:

$\forall c \in Advt_{in}$, $\exists d \in Query_{in}$,

s.t. match(c, d) != Fail.

Note that the order of *Query* and *Advt* has been transposed between the two expressions above. Suppose outQ ∈ Queryout and outA ∈ Advtout are two concepts. In case of output matching, the match(outQ, outA) function accepts outQ and outA as inputs and returns the degree of match between them. Four degrees of match are defined between a them:

- **Exact:** If outA is an equivalent concept to outQ or outA is a superclass of outQ. In case of a superclass relationship, it is assumed that the service provider has agreed to support *every* possible subclass of outA.
- **Plugin:** If outA Subsumes outQ. The relation between outA and outR is weaker as compared to the previous case since subsumption is indirectly inferred by the reasoner. It is assumed that the provider has agreed to support *some* sub-concepts of outA. We hence infer that outA can be *plugged in place of* the required outR.
- **Subsume:** If out Q Subsumes out A. The set of individuals defined by the concept, out A, is a subset of the set of individuals defined by the concept outR.
- **Fail:** If none of the above conditions are satisfied.

These four degrees as ranked as: *Exact* > *Plugin* > *Subsumes* > *Fail*. Here, x > y indicates that x is ranked higher (is a more desirable match) than y.

This matching condition can be included in the some existing algorithms that gives the better results so we have decided to use the best predicting algorithm that is the greedy approach for enhancing the semantic web services. The greedy approach is detailed below.

Greedy Approach: The algorithm adopts a greedy approach towards matching the concept-lists. For example, in the case of output matching, for each concept in the Queryout, it determines a corresponding concept in Advtout to which it has a *maximum* degree of match. Once all such max-matchings are computed, the minimum match amongst them is the *overall degree of match* between the *Query* and the *Advertisement*.

PROBLEM FORMULATION AND ALGORITHM

Following are some definitions that are define in our approach for formulating the composition problem:

- **Domain Ontology (O):** A Domain Ontology O consists of a set of concepts used to describe the domain in which a group of services is inserted. These concepts are related to each other according to subsumption relation, i.e. a concept $c_1 \in O$ subsumes another concept $c_2 \in O$ if $c_1$is a superclass of c_2; c_2 subsumes $c_1$if c_1 is a subclass of c_2;c_1 and $c_2$subsumes each other if they are the same.
- **Web Service (S):** A web service S is defined as $\{S_i, S_o\}$, where $S_i = \{S_1, S_2,\ldots, S_i\}$is a finite set of required input concept and $S_o = \{S_1, S_2,\ldots,S_j\}$ is a finite set of provided output concept.
- **Context Parameter (C_p):** Context parameter C_p is variable that is attached with web service. We use user context to fulfil its requirement.

- **Request (R):** A composition request R consists of a set of provided inputs $R_{in,}$ a set of required outputs R_{out}, and a context parameter C_p.
- **Service Repository (D):**A service repository D consists of a set of available services. Each service $S \in D$ is composed of a required inputs $S_i \subseteq O$,aset of provided outputs$S_o \subseteq O$,and a user context parameter C_p.

The Relationship between concepts of determined subsumption reasoning. So, given a required concept (*outR*) and an advertised concept (*outA*), the relationship between them can be labelled as follows

- **Exact:** If *outA* and *outR*are exactly the same concepts;
- **Plug-In:** If *outR*is subsumed by *outA*i.e. *outR*is subclass of *outA*;
- **Subsumes:** If *outR*subsumes *outA* i.e.*outR* is superclass of *outA*;
- **Fail:** Failure occurs when there is no subsumption relation between request and advertisement.

Subsumes: Two concepts c_1& $c_2 \in O$ can be related to each other in four ways. We define predicate *subsumes*: $(O \times O) \rightarrow$ {true, false} to express this relation as follows:

- *subsumes(c_1,c_2)* holds if only if $c_1$is generalization of c_2. (c_2 is then specialization of c_1)
- *subsumes(c_2,c_1)* holds if only if $c_2$is generalization of c_1. ($c_1$is then specialization of c_2)
- if neither*subsumes(c_1,c_2)* nor *subsumes(c_2,c_1)* holds, c_1 and c_2 are not related to each other.
- *subsume(c_1,c_2)* and *subsumes(c_2,c_1)* is true if and only if $c_1 = c_2$.

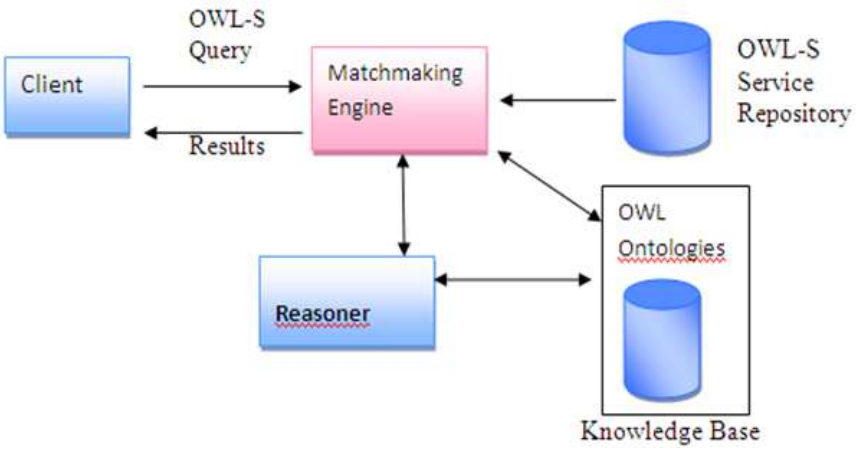


Figure 1: Proposed Framework

The operation of the framework can be described as follows:

A composition process begins when a Service Requester requests some functionality from service. Service Requester provides input parameters of service, required output and Quality of Service constraints as part of composition request.

Composition Engine starts to discover a set of services those provides required output and fulfils QoS constraints imposed by Requester from Service Repository with the help of Matchmaker.

Matchmaker module performs matchmaking operation between User's Query and Services provided by Repository. Reasoner is used with Matchmaker for computing similarity measure of the concepts. Matchmaker provides matched services to the Service Selection & Filtering module for selection, sorting and filtering of candidate services for composition. QoS parameters i.e. Throughput and Response Time are utilized for filtering purpose.The matchmaking process is performed with the help of domain ontology.

COMPOSITION ALGORITHM

In this section we propose algorithms for Semantic Matchmaking of Web services.

Algorithm 1

Following algorithm defines the matchmaking process by considering the user request and available services from the repository.

Input: R → Input Request, D → Service Repository

Data: X → the sorted list of composition to explore

Output: C → the solution composition, or ∅

1.Begin

2. ***foreach*** *outR* ∈ R_{out} ***do***

3. ***foreach*** *s* ∈ D | *c* ∈ s_o (*outR* subsumes *c*) ***do***

4. X ← match(out R, out A);

5. C_v ← { *s* };

6. ***end***

7. ***end***

8. ***while*** X ≠ ∅ ***do***

9. X ← *sort(descending, X, c)*;

10. C ← *popLastElement(X);*

11. ***if*** *isBest*(C) ***then***

12. ***return*** C;

13. ***foreach*** *c* ∈ *unknown(C)* ***do***

14. ***foreach*** *s* ∈ *promising(c)* ***do***

15. X ← *append(X, s ⊕C);*

16. ***end***

17. ***end***

18. end

19. return ∅

20. end

Algorithm Description

- In algorithm, line 1 to line 7 are used to search service by comparing requester output with repository services output and store that service in Cv. That also define function of matchmaking algorithm that call second algorithm of matchmaking conditions.
- Line 8 used to check whether the set X is empty or not. The set X is used to denote set of composition. The set C is used to store final resultant composition.
- Line 9 is used to sort compositions set X according to comparator function C_{cmp} and put them in descending order and store it in X.
- Line 10 is used to pop last composition from composition set X and store it in C.
- in line 11 and line 12 it will check that is C is best composition than it will return that composition.
- from line 13 to line 17 it will take input of composition C and match that input with output of service and services which match are appended to composition set x.
- Line 8 to line 17 are repeated until x ≠ Ø.

Algorithm 2

This algorithm defines the Degree of match for matching the requested services.

Input: outR (required concept), outA (advertised concept)

Output: Exact **or** Plug-in **or** Subsumes **or** Fail

1.	**if** *outR* = *outA* **then**
2.	return Exact
3.	**else if** *outA* subsumes *outR* **then**
4.	return Plugin
5.	**else if** *outR* subsumes *outA* **then**
6.	return Subsumes
7.	**Else**
8.	return Fail
9	**end if** **Algorithm Description**

	Take input as required concept outR and advertised concept outA.Check outR is equal to outA. If yes, then return exact condition. If no, then check outA is superclass of outR. If yes, then return plug-in condition. If no, then check outR is superclass of outA. If yes, then return subsumes condition. If no, then return fail condition.

FLOWCHART

Heuristic composition Algorithm depend on No. of service in repository so as we increases no. of service execution time is also increase. The time complexity in the worst case is $O(b^m)$

Where b is the maximum number successors of any nodes, namely, the number of services that can produce any concept required by the candidate composition, m is the maximum depth of the search space, namely, the maximum number of services in a composition.

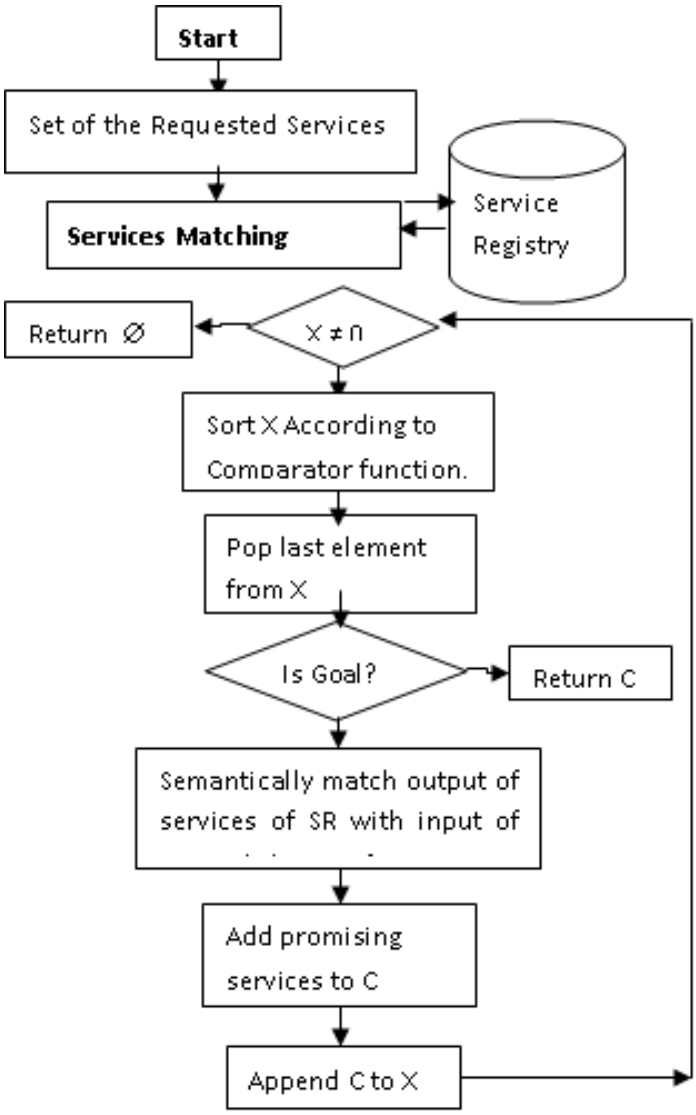


Figure 2: Flowchart of Composition Algorithm

EXPERIMENTAL SETUP

A. Web Service Challenge (WSC) Test Set

- The web service challenge started in 2005 to stimulate research into web service composition, growing and evolving each year.
- During the years 2005 to 2007, the Web Service Challenge focused on optimizing the service discovery and composition process solely, using abstractions from real-world situations. The taxonomies of semantic concepts as well as the involved data formats were purely artificial.
- Starting with the 2008 competition, the data formats and the contest data itself will be based on the OWL, WSDL, and WSBPEL schemas for ontologies, services, and service orchestrations to mimic real world scenario.
- In 2009, each service is annotated with nonfunctional properties. The Quality of Service (QoS) of a Web Service is expressed by values expressing its response time and throughput.

B. Hardware Configuration

All experiments were performed on a PC platform with a Intel(R) Core(TM) i3 CPU (2.40 GHz), Windows 7 Home Basic, and 3.00 GB RAM, 64 bit Opearating System All algorithms were implemented in Java with the use of the Eclipse tool.

C. Experiment Results with Comparison

This section defines the comparison of different matchmaking algorithms for execution time. This defines the Greedy approach, Bipartiate graph, Brute force method for matching the requested services.

Table 1: Different Matchmaking Methods Results

No of Services	Execution Time (ms)		
	Greedy Approach	Bipartiate Graph Matching	BruteForce Method
10	13	15	15
50	22	49	50
100	29	60	75
150	31	90	120
200	40	118	149
250	49	146	201
300	54	227	302

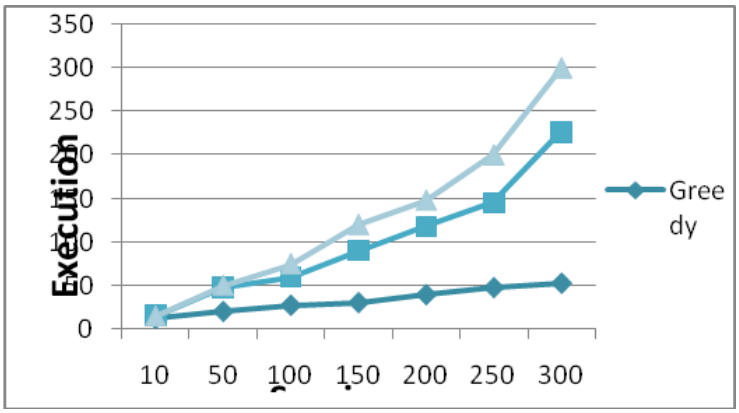
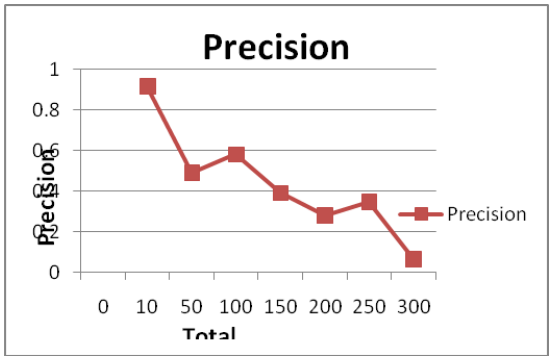


Figure 3: Comparison of No of Services Vs Execution Time



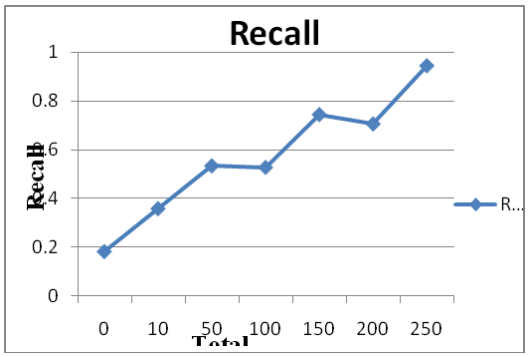


Figure 4: Comparison of No of Services Vs Precision **Figure 5: Comparison of No of Services Vs Racall Rate**

CONCLUSIONS & FUTURE WORK

We have identified that the heuristic approach also uses the semantic approach and in that we added semantic matchmaking approach for composition of the user requirement services. So semantic matchmaking can be added using the second algorithm that we proposed ahead in proposed work. In this we improved the matchmaking process by the use of the greedy approach. That shows the results that our greedy approach is far better than the other approaches for matching the queries.

To enhance the semantic matchmaking algorithm.To develop a prototype for the proposed work.To study and propose matchmaking algorithm for Cloud Services.

REFERENCES

1. W3C. SOAP 1.2 Working draft, 2001. http://www.w3.org/TR/2001/WD-soap12-part0- 20011217
2. UDDI Consortium. UDDI Executive White Paper, Nov. 2001.
3. T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American May 2001
4. E. Christensen, F. Curbera, G. Meredith and S. Weerawarana., Web Services Description Language (WSDL) 1.1, 2001.
5. S.McIlraith, T.C.Son and H.Zeng., “Semantic Web Services”, IEEE Intelligent Systems, 16(2), Mar.2002.
6. Antonio Bucchiarone 1, Stefania Gnesi, “A survey on Service Composition Languages and Model”, International Workshop on Web Services Modeling and Testing WS-MaTe 2006.
7. G.Alonso, F.Casati, H.Kuno and V.Machiraju., “Web Services: Concepts, Architectures and Application”, Springer-Verlag Berlin Heidelberg 2004.
8. M.Dean, D.Connolly, F.van Harmelen, J.Hendler, I.Horrocks, D.L.McGuinness, P.F. Patel-Schneider and L.AStein., “Web Ontology Language (OWL) “ Reference Version1.0, W3C Working Draft 12 November 2002.
9. Schahram Dustdar, Wolfgang Schreiner, “A survey on web services composition”, International Journal of Web and Grid Services, Vol. 1, No. 1, 2005.
10. Jorge Cardoso, “Quality of Service and Semantic Composition of Workflows”, PhD thesis, Department of Computer Science, University of Georgia, Athens, GA (USA), 2002.

11. Zeng Liang zhao，Benatallah B, Dumasm, et al., “Quality driven Web services composition”, in Proc. of the 12th International Conference on World Wide Web. New York: ACM Press, 2003: 411-421.
12. L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang., “QoS- aware middleware for web services composition”, IEEE Transactions on Software Engineering, 30(5):311–327, May 2004.
13. F. Lautenbacher, B. Bauer, “A Survey on Workflow Annotation and Composition Approaches”, In M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, N. Stojanovic (eds.) Semantic Business Process and Product Lifecycle Management, in Proc. of the Workshop SBPM 2007, Innsbruck, CEUR Workshop Proceedings, ISSN 1613-0073, online CEUR-WS.org/Vol-251, April 7, 2007.
14. Kaouthar Boumhamdi, Zahi Jarir , “A Flexible Approach to Compose Web Services in Dynamic Environment”, International Journal of Digital Society (IJDS), Volume 1, Issue 2, June 2010.
15. George Baryannis and Dimitris Plexousakis, “Automated Web Service Composition: State of the Art and Research Challenges”, Technical Report ICS FORTH/TR-409 October 2010.
16. Dongsong Zhang, Minder Chen, and Lina Zhou, “Dynamic and Personalized Web Services Composition in E-Business “, ISM Journal 2005.
17. Bansal S Vidal J M. “Matchmaking on web services based on the DAML-S service Model”. In Proc. 2nd Int. Joint Conference. Autonomus Agents and MultiAgent System, Melbourne, Australia. July 14-18,2003.
18. Klein M. Bernstein A.“Semantic services on the semantic web using process ontologies”. In Proc Int. Semantic Web Working Symposium(SWWS), Amsterdam: IOS Press 2001.
19. Jaeger M C Rojec-Goldman G, Liebetruth C et al, “Ranked matching for service description using OWL-S". In Proc. Communication in Distributed System, Kaiser Saulter, Germany 2005.
20. Verma K Shivashnmugam K, Sheth A et al, Meteor S-Wsdi: “A scalabel infrastructure of registries for semantic publication and discovery of web services”, Jouranl informational Technology and Management 2005.
21. Bentallah B Hacid M, Alein et al. “On automatic web service disccovery”, VLDB Journal 2005.
22. Frederico G. Alvares de Oliveira Jr., Jos´e M. Parente de Oliveira, “An Infrastructure for Evolving Dynamic Web Services Composition”, International Journal of Intelligent Computing Research (IJICR), Volume 1, Issue 1/2, March/June 2010
23. Umesh Bellur, Harin Vadodaria and Amit Gupta, “Semantic Matchmaking Algorithms”, Advances in Greedy Algorithms, Book edited by: Witold Bednorz, ISBN 978 953-7619-27-5, I-Tech, Vienna, Austria, pp. 586, November 2008.
24. Bin Xu, Po Zhang,Juan-Zi Li, Wen-Jun Yang, “A Semantic Matchmaker for Ranking Web Services”, J. Computing Sci. & Technology, July 2006 , Vol 21, No 4, pp 574-581.
25. Umesh Bellur and Roshan Kulkarni. "Improved matchmaking algorithm for semantic web services based on bipartite graph matching Web Services". ICWS 2007, 88--93
26. S. Agarwal and R. Studer, "Automatic Matchmaking of Web Services," Proc. Int',l Conf. Web Services (ICWS

',06), pp. 45-54, Sept. 2006.

27. Gao Shu, Omer F. Rana, Nick J. Avis, Chen Dingfang "Ontology-based semantic matchmaking approach", Advances in Engineering Software, Volume 38, Issue 1, Pages 59-67, January 2007. www.sciencedirect.com, www.elsevier.com/locat/advengsoft
28. Giuseppe Fenza, Vincenzo Loia, Sabrina Senatore "A hybrid approach to semantic web services matchmaking", International Journal of Approximate Reasoning 48 (2008), Computer and Information Technology (CIT), 2012 IEEE 12th International Conference , 147 - 151 ,Oct. 2012. www.sciencedirect.com , www.elsevier.com/locate/ijar
29. Jing Li "A Fast Semantic Web Services Matchmaker for OWL-S Services", Journal Of Networks, Vol. 8, No. 5, May 2013, Academy Publishe.
30. Ulrich Lampe, Stefan Schulte, Melanie Siebenhaar, Dieter Schuller, Ralf Steinmetz, "Adaptive Matchmaking for RESTful Services based on hRESTS and MicroWSMO.", Walter Binder and Heiko Schuldt: Proceedings of the 5th Workshop on Enhanced Web Service Technologies (WEWST 2010),The Association for Computing Machinery (ACM), p. 10-17,December 2010.
31. Stephen S. Yau and Junwei Liu, "Functionality-Based Service Matchmaking for Service-Oriented Architecture", *ISADS, page 147-154. IEEE 2007*.
32. Bo Jiang, Zhiyuan Luo "A New Algorithm for Semantic Web Service Matching", Journal Of Software, Vol. 8, No. 2, February 2013.
33. Kim Christensen, Thorbjørn Højgaard Olesen, Lone Leth Thomsen," Matching semantically described web services using ontologies", ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, Vol.35, No.3A, 2006
34. Tanveer Syeda-Mahmood, Gauri Shah, Rama Akkiraju, Anca-Andrea Ivan,Richard Goodwin," Searching Service Repositories by Combining Semantic and Ontological Matching", Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference, 13 - 20 vol.1, 11-15 July 2005